Designing a Cloud-Based Infrastructure for Spectrum Sensing: A Case Study for Indoor Spaces

Ayon Chakraborty and Samir R. Das
Computer Science Department, Stony Brook University, Stony Brook, New York 11794, U.S.A.
{aychakrabort, samir}@cs.stonybrook.edu

***Abstract*—We argue that spectrum sensing on mobile clients will be both necessary and feasible if we wish to manage the white space spectrum optimally in indoor spaces. We demonstrate the necessity with a set of empirical measurements showing the need for fine grained sensing. We demonstrate the feasibility by building a spectrum sensing infrastructure that collects measurements from sensing devices to analyze and better use spectrum resources. The infrastructure consists of mobile spectrum sensors that are built using DTV receiver dongles interfaced with Android-based mobile devices and a cloud-based central server to manage such sensing devices. We also show results about resource consumption (energy, network overhead) involved in operating such sensors. The vision is ultimately creating a system where mobile devices perform part-time spectrum sensing in a coordinated fashion under the control of a central spectrum manager. We lay out the research challenges based on our initial prototyping and benchmarking experience.**

***Keywords*-mobile spectrum sensing, white space, dynamic spectrum access**

I. Introduction

It is widely anticipated that the use of white space (WS) spectrum will dominate in indoor spaces [20]. The purported reason is that almost 70% of all mobile use is indoors [26] and one of the greatest use of white space is offloading from licensed cellular bands [20]. A recent measurement study based on Hong Kong [26] also shows that there is a significant improvement in availability of TV white space (TVWS) spectrum indoors relative to outdoors. This provides more hope for indoor use. In general, there is already some belief that indoor use could be the only 'sweet spot' for exploiting TVWS [20]. The reason for this is that outdoor TVWS availability is known to be poor in densely populated areas [15]. [1] This scenario is not likely to change with the new white space spectrum beyond the TV bands.[2] While the above observations collectively indicate that the understanding and using indoor WS should be of critical importance, with the exception a few recent studies [26] very little investment has been made to study indoor WS.

However, indoor use brings in challenges in spectrum management. Several papers suggested that WS networks should do some form of centralized spectrum management for the secondary devices [24]. Though no clear architecture has yet emerged we posit that a centrally computed 'radio environment map' (REM) (estimate of power spectral density at all points in the useful space) is an important tool for such spectrum management. But computing REM can be challenging indoors. Empirically-based propagation modeling is not reliable indoors. Deterministic models like ray tracing needs high resolution inputs about the indoor environment that may be hard to obtain in practice. Thus, a measurement-based approach is likely the only possibility. Such measurements have indeed been proposed, but only in the APs [24] (assuming an 'cellular' WS architecture). This is generally easy to do as the APs have abundant power budget and adequate backhaul to a central spectrum database/manager.

However, as we will demonstrate later that even in a small cell-like setup AP-only sensing cannot estimate the REM within the needed degree of accuracy. This is again related to the vagaries of indoor propagation. The only recourse will be more granular sensing, perhaps via an additional set of dedicated spectrum sensors [26]. But this may not be a cost-effective choice always.

A. Mobile Spectrum Sensing

We propose an alternative where spectrum measurements are also performed in a coordinated fashion on spectrum sensors that are either add-on or integrated into mobile devices [17], [22]. These include any mobile device from phone or tablet to a wearable platform such as glass or watch. While the general concept of distributed, coordinated spectrum sensing is hardly new, much of the existing work has been limited to laboratory grade spectrum analyzers or powerful SDR-based spectrum sensing [16].

Modern mobile platforms provide an attractive alternative. Many provide multicore processors and GPUs to provide enough compute power. They are also everywhere as a significant and growing fraction of humanity possesses one or more of these devices. Further, they are often idle. They do lack appropriate radio interfaces for white spaces at this time. But obviously they will acquire such interfaces if they are to use white space spectra. The technical feasibility of mobile-based spectrum sensing or cognitive radio platforms has already been demonstrated [17], [22], [11], [19], [28]. Also, while not directly related to WS, [23] has demonstrated the utility of client-assisted monitoring of wide-area wireless networks

[1] A quick look at Spectrum Bridge database [7] at the time of this writing shows only 0, 3 and 5 TVWS channels available in most locations in New York City, San Francisco and Chicago, respectively!

[2] For example, naval radar bands that are being considered for deregulation [20], [10] likely will suffer from similar issues as roughly 40% of US population lives in coastal shoreline counties [2].

2325-2944/16 $31.00 © 2016 IEEE
DOI 10.1109/DCOSS.2016.28

for improving the operators' understanding of network performance.

B. Challenges

There are, however, many challenges. They fundamentally center around two issues: (i) architectural design and (ii) understanding and managing the accuracy vs resource use tradeoffs:

Architecture: An end-to-end framework must be developed such that spectrum data will be collected from numerous sensors and collated at the backend (a central perhaps cloud-based 'spectrum manager') to create the REM. In our case, REM may need to be represented probabilistically as the spectrum measurements will at best be samples taken at discrete points in time that would present a distribution of measured radio signal power at specific frequencies at that location.

Accuracy: It is unclear whether inexpensive radios integrated onto a phone and with small, built-in antenna can perform sensing with the needed accuracy. Also, due to resource limitations, the sensing can be sparse in frequency, time and space, and also the data could be compressed.

Resource: While mobile devices are often idle and may have ample processing power, they are energy poor. The radio front end can also have limitations in the sense of tunability and sampling rate. Thus, spectrum sensing must be controlled by the central server for optimally exploiting the radio capability for a given power budget. The cost of network connectivity is also an issue. The spectrum data may need to be compressed before transmission or may have to be analyzed partially (e.g., the FFT on the I/Q samples can run on the mobile and only differences are communicated to the server, etc.). Considering the energy and backhaul costs, the server may need to schedule sensing tasks on the mobiles carefully.

In this paper, we limit ourselves to only parts of these challenges. We make the following contributions:

- We develop a prototype platform using commodity devices – DTV receiver dongle interfaced with an Android-based mobile device (Figure 3). We also demonstrate a cloud infrastructure about handling such sensing modules (Section III).
- We perform preliminary spectrum sensing measurements to generate radio environment maps and analyze their accuracy to show the power of client-assisted sensing (Section IV) in indoor environments.
- We highlight the resource consumption issues (Section V) to further demonstrate the practicability of such a system.

II. Related Work

Recent literature focuses on accurate identification of TV Whitespaces. [26] introduces WISER, a system which identifies and tracks indoor white spaces in a building, without requiring user devices to sense the spectrum. Whereas [27] proposes an architecture where spectrum sensing is performed in city-wide public transport. But all these works try to search for available whitespace (i.e., absence of primary signals).

The whitespace channels being known the next logical step to follow is how best to utilize those channels. To the best of our knowledge there is no prior work which addresses the whitespace channel allocation problem and proposes a system for doing the same. A similar approach as ours is taken in [23] where client-assisted monitoring of wide-area wireless networks is done to assist network operators to better understand performance characteristics.

Recently, there have been several initiatives to build low power, mobile spectrum sensors [13], [11], [19], [28] prototype. However, the focus of our work is to build a cloud infrastructure on top and laying out the challenges in doing so.

III. Cloud Infrastructure for Spectrum Sensing

One of the main bottlenecks in building such an infrastructure lies in creating a mobile TVWS sensor. At the current time, experimental prototyping of a TVWS sensor on a mobile device is hard. While current generation smartphones have multiple radios (WiFi, 3G/4G, Bluetooth, NFC, etc) they do not operate in the TV band. In general, an interface capable of tuning to arbitrary frequencies in the operating band and obtain I/Q samples is absent; so we cannot even use another band as a proxy for TVWS. Past work has considered developing custom, small form-factor boards with RF front ends, ADC and FPGA that can be interfaced with the phone [17]. Chip-level design of the wideband frontend of an SDR receiver has also been demonstrated [22]. In general, there is a growing interest in developing micro-SDRs with small form-factor and energy budget [18], [14]. However, these are research prototypes and are not widely available.

Instead, we rely on commodity devices to understand the potential of our techniques. We use a commodity Digital TV receiver dongle (DTV receiver with an USB interface with the form factor of a thumb drive) that has a specific chipset (Realtek RTL2832U demodulator) with the ability to export I/Q samples [5]. Such dongles can be used as inexpensive software radio receivers as the demodulator (RTL2832U) supports a debugging mode where it passes the I/Q samples directly from the tuner's ADC to the device. The dongle is provided with an aerial TV plug and is interfaced to a mobile device via the USB port. This forms a miniature spectrum sensor. We have connected a low gain antenna to the dongle. However, in principle such antennas can be built inside the phone as past work has shown [17] [22]. In the following, we describe the hardware prototype we build followed by the cloud based sensing infrastructure.

A. Mobile TVWS Sensor Hardware Prototype

Smartphone / Tablet: We have used the EzCap Digital ATSC TV dongle. The dongle uses a Rafael R820t tuner [8] supporting the ATSC standard. The tuner works with DVB-T, ISDB-T and DTMB standards as well. Figure 3 shows our prototype spectrum sensors. The dongle has a USB 2.0

(a) RTL Dongle with smartphone (b) RTL Dongle with tablet (c) RTL Dongle with Raspberry-Pi

Fig. 1: Some examples of platform configurations considered in this work.

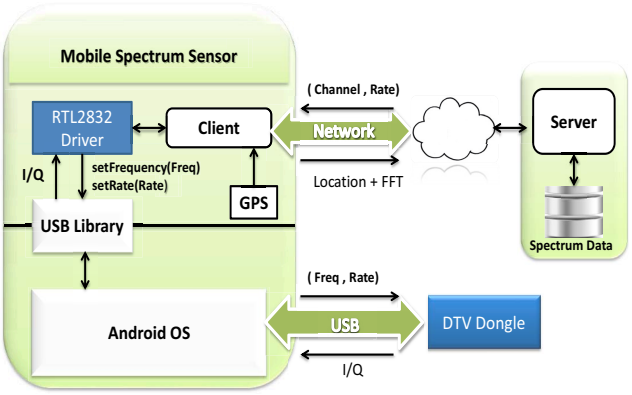


Fig. 2: A system overview of the prototype mobile spectrum sensor.

(a) TV signal detected (b) Potentially white space

Fig. 3: Snapshot of our android sensing client application. The left figure shows the TV pilot while the right one doesn't for two different channels.

interface and can be interfaced to any mobile device that supports USB OTG (on-the-go) so that it can host a USB accessory. For prototyping purpose we used three types of devices: smartphones, tablets and raspberry-pi.

Raspberry-Pi: The other prototype we build was using a Raspberry-Pi embedded computer. The Raspberry-Pi was powered externally and an USB WiFi dongle was attached to it for providing network connectivity. We kept ourselves limited to Raspberry-Pi platform only for proof-of-concept, however other embedded computing platforms like Beagle-Bone, Panda board etc. are also good candidates for creating a mobile spectrum sensor prototype.

B. Measurement Infrastructure

In the following we describe different components of our measurement infrastructure.

Spectrum Data Broker: At the core of our infrastructure sits the *spectrum data broker* that facilitates message passing among different components in the system. The broker uses MQTT protocol [4] that follows a publish/subscribe messaging pattern. MQTT being one of the very successful messaging protocols in the Internet-of-Things domain was our natural choice over HTTP. Unlike HTTP where the server needs to interact with all clients independently, the MQTT server (or client) can *'publish'* a message under a certain *'topic'* and all MQTT clients who have *'subscribed'* to that particular topic receives the message. MQTT has built-in support for QoS unlike HTTP. HTTP takes up more bandwidth (due to the text-based nature, headers etc.) than MQTT. Apart from that MQTT has much less CPU/memory footprint (less energy hungry) compared to HTTP. Such features motivate us to adopt MQTT protocol in our platform. We use HiveMQ [3] as our spectrum data broker.

Sensing Application: The Open Source Mobile Communications (Osmocom) [5] has a community-supported project developing software support for using DTV dongles as above as functioning SDR receivers. We ported the existing Linux libraries and drivers for such dongles to the Android and the Raspbian platform.

The application runs on the mobile spectrum sensor. It uses a MQTT client service that subscribes to a topic: *'scan'*. All spectrum sensing requests that are sent (by the sensing

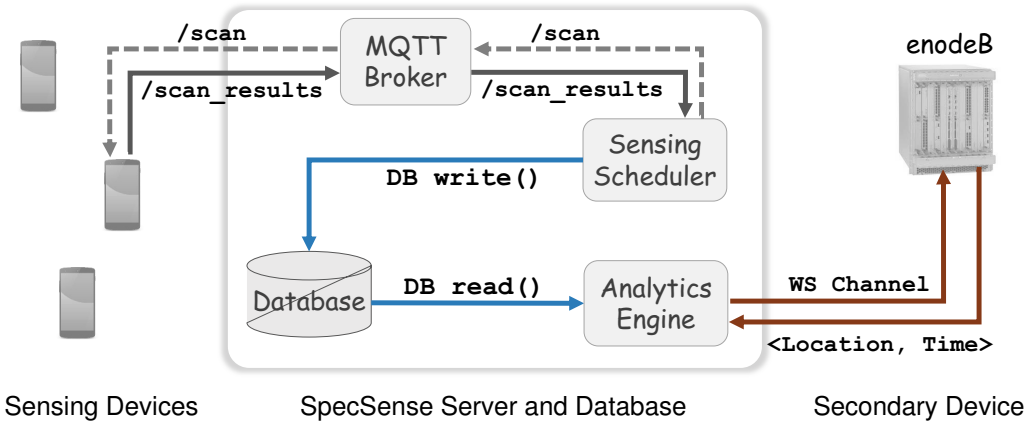


Fig. 4: Software architecture of our mobile spectrum sensing infrastructure. The secondary device can be a enodeB or an WiFi access-point that asks for a white space channel from the system.

scheduler) under the *'scan'* topic are received by this client and a scan is issued. The message contains information about the scan operation like the channel number, sample rate, sensing duration and the number of samples. The application obtains the raw I/Q samples from the sensor hardware, performs an FFT operation on it and subsequently computes the power spectral density (PSD) over the sensed channel. It then publishes the scan results under the topic *'scan_results'* and the sensing scheduler that subscribes to this topic receives it. The scan results are also accompanied by a location stamp and the battery state of the mobile device. Additionally the application periodically updates the scheduler with its location and battery power.

Sensing Scheduler: This issues the scan requests to the mobile spectrum sensors. Currently we our system supports three types of scheduling algorithms.
Random: We select the sensors randomly and assign a random DTV channel (14 – 51) for it to scan.
Geographical Loadbalancing: We partition the entire area into discrete grids and try to maximize the number of grids covered by sensors that are scheduled to scan.
Frequency Loadbalancing: In this case we try to divide the sensing tasks based on the channel occupancy patterns. Some channels change occupancy too often and needs frequent scanning and the others require less frequent scanning.

In all such cases we try to improve the cost–performance tradeoffs in a large deployment setting.

Spectrum Database: The scheduler enters the information in a SQL database. We also provide a web front-end and data analytics engine over the database. The details are not provided here for brevity. The current deployment spans across our university campus ($\approx$ 6 sq. km) with $\approx$10 mobile spectrum sensor nodes.

We demonstrate the feasibility of building a mobile spectrum sensor combined with a cloud infrastructure to house such sensors and collect related sensing data. In the following we show an important indoor use case of channel selection where such a system can play an important role.

IV. Experiments

In this section our goal is to provide an empirical demonstration of the power of client-assisted sensing in the indoor context. To do this, we will perform extensive indoor spectrum sensing experiments followed by performance analysis.

A. Experimental Setup

Emulating Small Cells: Our setup emulates indoor small cells operating in UHF frequencies. Since our prototype spectrum sensor doesn't have a UHF transmit interface, we use Ubiquity XR7 wireless cards [9] (operating in 760-780MHz band) as secondary devices [3]. These cards are mounted on embedded processor boards (Avilla GW2348 [1]). 14 such nodes are deployed around our department building (in the same floor) covering approximately an area of 300ft x 150ft. For the purpose of demonstration we limit ourselves to 2 channels (each 5MHz wide, henceforth we call them channels X and Y) and configure the nodes to operate in either of them. We make sure that no other interfering signal is present in those channels. The center frequencies of X and Y are kept 20 MHz apart to prevent adjacent channel interference influencing the results. 4 nodes are configured such that they mimic WS APs and drive their own 'small cells'. Two of these APs operate in channel X while the other two in channel Y. Additionally, each AP node is also equipped with a regular

[3]The 700MHz band (Channels 52-69) is not a part of the current DTV spectrum, although such was the case before this band was auctioned in 2008. See: http://fjallfoss.fcc.gov/edocs_public/attachmatch/DA-07-3415A1.pdf

802.11 interface in the 2.4 GHz band that connects to a Central Server. This serves as the backhaul and is used to communicate control instructions (channel, bandwidth, transmit power etc) to the AP. The rest of the nodes (10 of them) mimic clients and connect to the 4 APs. We created a constant UDP traffic at 6Mbps between the clients and the APs. The output power of the antenna in the clients is kept at 15mW, while that in the AP is 27mW. We do this to introduce secondary interference in the cells.

Sensing Probes: The node (both the whitespace clients and APs) are not equipped with hardware capability to sense spectrum by itself. We use our sensor prototype to sense the spectrum on its behalf, keeping it at the same location with the node. Such an arrangement (node combined with the sensor) augments the sensing functionality of the node. The sensors monitor both the channels and report the sensed data (in form of a 1024-bin FFT) to the central server. The FFT data is location stamped based on the node-id, and it is assumed that the server is aware of the location of the nodes with respect to a reference coordinate system.

Central Server: As described before, the central server collects location-stamped sensing data from the sensors and generates the REMs for both channels separately. Next, it uses these two REMs to identify the channel that is best for each AP. Clearly, there could be many strategies for channel selection. In the current system we consider the average interference level per cell for a given channel as a metric to identify the best channel. In this setup, we do not have dynamic/mobile secondaries so the channel allocation does not change with time. We aim to take it up as a future work. In case of dynamically changing REMs, the central server instructs the APs to switch channels.

REM Generation: Recently, there has been several studies in employing standard methods of spatial statistics in mapping the radio environment (see, e.g., [25]). Empirical measurement based work has also been done using similar approaches [21]. We take the same basic approach (Ordinary Kriging) by assuming the radio signal strength at each frequency to be a random field in 2D that is sampled at some random (spatial) intervals. In this experiment our goal is to create REMs for different number of sensing locations.

B. Results

Recall that our testbed has four APs and ten clients operating at two possible UHF channels. We perform spectrum sensing (for both channels) at all nodes and study how a sparser sensing would impact REM, channel selection and overall performance.

REM Accuracy: Here, we study how the REM's accuracy is impacted by sparser sensing locations. First, we sense the spectrum at all 14 locations where a node is present and compute the REM. Of course, this is the best we can do in terms of client assisted sensing and hence we consider this as representing ground truth. Next we decrease the number of sensing locations and create a REM for each instance. Figure 5 shows the REM generated by AP-only sensing compared to the ground truth. For a given point in the REM, the absolute difference in the estimated signal strength and the ground truth is the estimation error, which is calculated for all gridpoints on a 300x150 grid (i.e., each grid cell is 1ft x 1ft) overlaid on the REM. Figure 6 shows the CDF of REM estimation errors for different number of sensing locations. Note that AP-only sensing provides a median error of approximately 12 dB. This obviously represents a significant estimation error. A visual representation of this is in Figure 5 that shows the resultant REMs as heat maps.

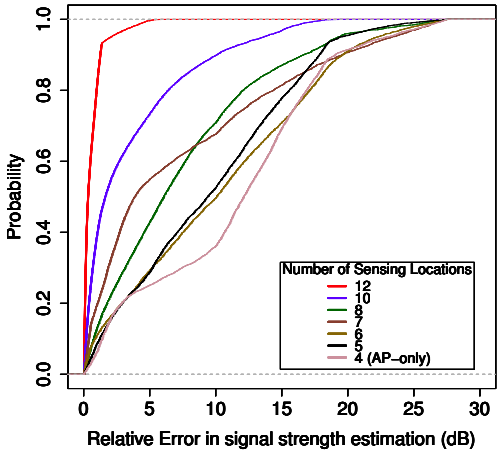
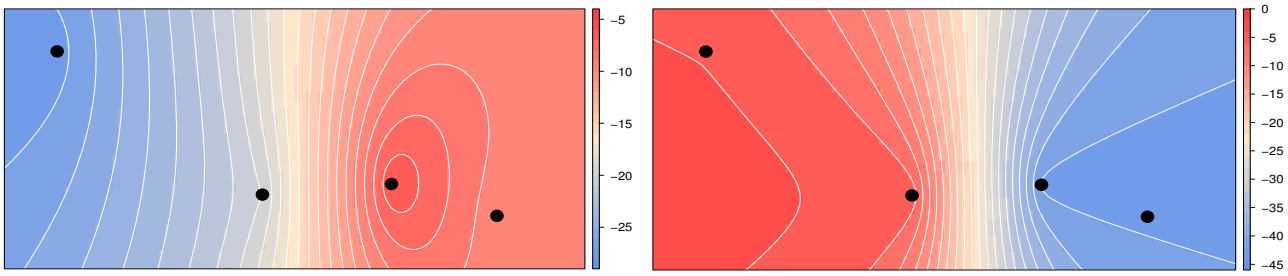


Fig. 6: CDFs of the estimation errors in the REMs with different number of sensing clients.

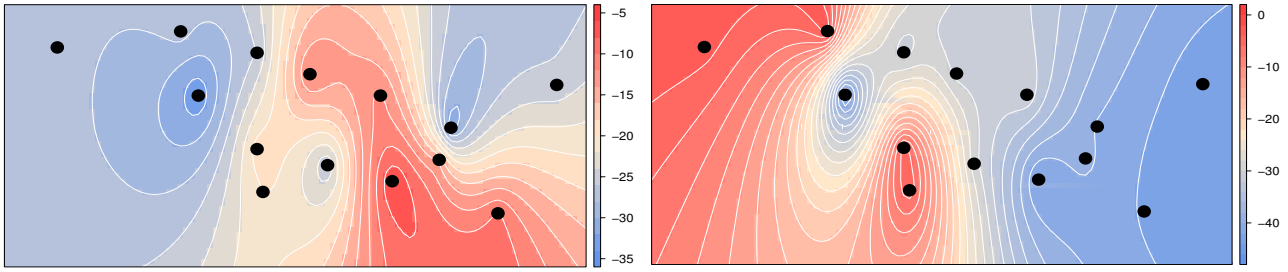
As expected, error improves with more clients performing the sensing (Figure 6) falling within few dBs of median error with 8+ clients sensing. However, some parts of the REM (top 20% in the plot) remain fairly inaccurate even with 10 clients sensing.

Channel Choice: Estimation errors of REM is just one part of the issue. What essentially matters is whether a poor estimation leads to a poor choice of channels that impacts performance, For example, in Figure 5 for about 25% cases on average the best channel as derived from the Figure 5(a), i.e., AP-only disagrees from that derived from 5(b), i.e., when all clients are sensing. Interestingly these 25% locations are clustered among a handful regions. Furthermore, our experiments represent one of the many scenarios possible in indoor environments. Situations can be more complicated with mobile secondaries with variable transmit powers. Also, larger deployments, say across multiple floors/campus-wide areas, such inaccuracies can be more significant.

Performance: It is also interesting to quantify the loss in performance due to the choice of an inappropriate channel. We do the following experiment in order to evaluate such loss. For locations where the choice of best channel differs between the two REMs (based on AP-only and Client-assisted sensing respectively), we introduce a transmitter-receiver pair and operate them in both channels consecutively. The transmitter



(a) REM based on AP-only sensing (two channels).



(b) REM based on Client-assisted sensing (two channels).

Fig. 5: REM based on AP-only and client assisted sensing are shown for both the channels. The left-side plots correspond to channel X while that on the right-side correspond to channel Y. The dots in the figure represent sensing locations.

sends a constant bit-rate UDP traffic at 6 Mbps and we measure the observed throughput and RTT (ping latency) at the receiver end. We repeat the same experiment for the two channels. Figure 7 compares the CDFs of throughputs and RTT (measured every second for approximately 10 minutes) for a typical such location. Note that the median value for throughput and RTT differs by more than 50% and 100% respectively. Across different such locations we hve observe a similar trend.

C. Outdoor Case

While we have concentrated specifically indoors and secondary sensing in this work, the general approach is obviously applicable outdoors and/or primary sensing as well. Recent measurement experiments [26], [27] have shown that the spectrum databases are way too conservative in identifying WS. There is already some interest in improving inaccurate spectrum databases by augmenting it with actual spectrum sensing measurements. For example, [27] takes a wardriving approach where such sensing is performed in city-wide public transport. While the technology for doing this is immediately feasible, this approach is limited in scope as it can provide measurements only from certain limited areas. It remains to be see whether this can adequately serve the purpose of identifying WS in locations where people frequent. On the other hand, the proposed approach of client-assisted sensing can provide unlimited options regarding measurement locations. We have reported preliminary outdoor measurements using the same platform in [13].

No. of Bins	Time/FFT (ms)	Energy/FFT (mJ)
256	11.3	5.73
512	26.4	13.32
1K	60.2	30.18
2K	139.8	70.6
4K	334	167.3

TABLE I: Computation time and energy overheads for different FFT bin sizes according to PowerTutor measurements.

V. Resource Usage

Much of the client-assisted spectrum sensing is to be done on resource constrained mobile devices (e.g., smartphones, tablets, wearable smart devices such as glass or watch). A natural question that arises is how well the system performs in terms of resource usage. In this section we address the different components related to resource (energy, CPU, network) consumption in the mobile client. At the core of such mobile client sits an application which receives scheduling instructions from the central server, issues scan instruction to the radio, computes the FFT based on the I/Q samples and communicates back the sensed data as necessary. Broadly there are three different components associated with such a system, viz., a) Sensing (powering the radio and fetching I/Q samples), b) Computation (primarily FFT computation based on the samples), and c) Communication (uploading the sensed data to the spectrum manager and receiving various control information, e.g., related to scheduling).

Sensing: We have measured the current flowing into the dongle connected to the tablet's USB OTG bus at 5V. The current draw is measured in two states: Idle (when the dongle

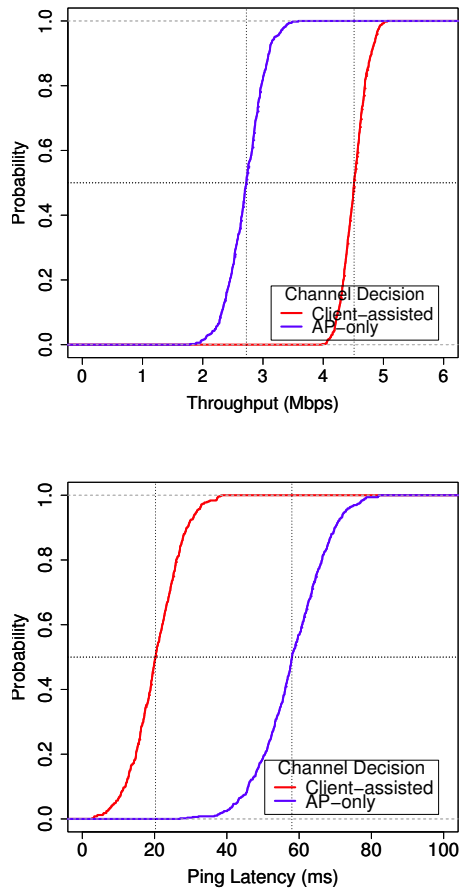


Fig. 7: CDFs of throughput and ping latencies at a given location for using the best channel chosen by AP-only sensing versus Client-assisted sensing.

is attached to the tablet, but does not perform scanning) and Scanning (the dongle is actively sensing spectrum). We have not noticed a very strong sensitivity to the sampling rate, but the reported measurements are for 2 Msps sampling rate only. On an average, the idle state draws about 90 mA while 160 mA is drawn in the scanning state. The maximum variation of the measured current value in in either case is within 5mA. Thus, the power consumed by the dongle in scanning state is about 800 mW. We further note that the idle state power consumption can be completely eliminated by powering off the USB bus from the application software.

Computation: The application software that controls the client's system has a computational overhead. The application accomplishes several tasks including controlling the dongle, fetching samples, FFT computation and some intelligent decision making based on the scan results. Among these tasks, FFT computation is the most CPU intensive. So, we evaluate its energy overhead on our prototype tablet-based platform described before. The PowerTutor App [6] is used to profile the CPU energy consumption application for different FFT bin sizes. The results are shown in Table I. Summarizing from the table, FFT consumes roughly 500mW of power regardless of the bin size. The CPU load for the application while it is actively scanning and performing the FFT on the samples is approximately 42%.

Communication: The communication interface in the mobile client has a two fold use. First, the central server probes the client to schedule a scan operation. Second, the client conveys the scan results (FFT) to the server. Note that the scan result is only a few KB data. Additionally, not every scan result has to be communicated, i.e., communicate only if the scan results differ by a predetermined threshold. Currently our prototype uses the WiFi interface to communicate the scan results to the server.

To provide the reader with some calibration we review how this energy usage compares with typical smartphone applications. A recent study [12] has performed a suite of benchmark measurements for typical power draw for different activities (e.g., phone call, web browsing, video etc.). For example, web browsing can consume around 500 mW while phone call and video consume approximately 800 mW and 500 mW, respectively. Again these measurements discount the screen backlight which alone consumes several 100s of mW (depending on screen size, brightness and technology). In comparison, assuming one FFT operation (1K bin size) per second and associated sensing (1K I/Q samples) and backhaul communication (over WiFi) our prototype is estimated to consume less than 1000 mW on average for spectrum sensing. (This analysis uses the measurements above. The energy needed for WiFi communication is separately estimated for the same device.) This is not significantly higher than typical applications given that there is no added energy cost for the screen.

Also note that a major component of the energy was spent in powering the sensor's hardware. With the availability of WS interfaces in secondary devices, we presume that low power on-chip radios will be integrated with the device which will reduce energy usage. Further, various compression techniques can be used for the communication part, e.g., only differences can be communicated and that too only when they are larger than a threshold, etc.

Scheduling Sensing Tasks: It is clear that continuous sensing on all clients at all times is not resource efficient. This gives rise to a significant task scheduling problem that we will pursue in our future work. The idea is to estimate the 'utility' (in terms improvement in REM accuracy) coming from one unit of sensing on a specific client. If this utility is significant relative to the resource 'cost,' this client can be tasked for sensing. The resource cost is dependent on the 'context' of the client that includes, among other things, the remaining battery power. The information gain is tied to the the location of the client, whether the client is mobile, etc. that are part of the context as well. Historical information can be mined by the server to ascertain the utility of sensing. The sensing can be broken down further in frequency dimension. For example, it

may be more beneficial for certain clients to sense only certain portions of the WS spectrum.

VI. Conclusion and Future Work

In this work we have argued that spectrum sensing on mobile clients will be both necessary and feasible if we wish to manage the WS spectrum optimally in indoor spaces. It is necessary as indoor propagation environment is complex enough that modeling-based methods will not be adequate. On the other hand, using a DTV-based dongle as a proxy for a spectrum sensor we demonstrated the feasibility of spectrum sensing on an Android mobile platform and a Raspberry-Pi device. Second, we built a scalable and robust system to collect data from such sensors and implemented several scheduling algorithms to collect such data. Initial validation experiments show the power of client-assisted spectrum sensing in getting an accurate rendition of the radio environment that in turn impacts performance. In this work, we have laid out the vision of such systems, presented a prototype implementation and performed preliminary validation experiments. The entire gamut of challenges mentioned in the paper forms parts of our ongoing and future work. In particular, we are pursuing developing sophisticated algorithms to address the scheduling scanning task scheduling problem by means of crowdsourcing and realizing it in a real system.

References

[1] Avila GW2348-4. http://site.microcom.us/gw2348-4ds1-3.pdf.

[2] Coastal Population. http://coastalchallenges.com/2010/01/31/un-atlas-60-of-us-live-in-the-coastal-areas/.

[3] HiveMQ Enterprise MQTT Broker. http://www.hivemq.com/.

[4] MQTT Protocol. http://mqtt.org/.

[5] The open source mobile communications (Osmocom), SDR project. http://sdr.osmocom.org/.

[6] PowerTutor App. http://ziyang.eecs.umich.edu/projects/powertutor/.

[7] Spectrum Bridge website. http://spectrumbridge.com.

[8] Terrestrial DTV silicon tuner. Rafael Microelectronics, Inc. http://www.rafaelmicro.com/downloads/R820T.pdf.

[9] Ubiquity XR7 Cards. https://dl.ubnt.com/xr7_datasheet.pdf.

[10] Second report and order and memorandum opinion and order in the matter of unlicensed operation in the TV broadcast bands. FCC ET Docket 08-260, Nov. 2008.

[11] R. Calvo-Palomino, D. Pfammatter, D. Giustiniano, and V. Lenders. A low-cost sensor platform for large-scale wideband spectrum monitoring. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 396–397. ACM, 2015.

[12] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *In* USENIX, 2010.

[13] A. Chakraborty, S. R. Das, and M. Buddhikot. Radio environment mapping with mobile devices in the TV white space. In *In* Proc. ACM Mobicom*, (Poster)*, 2013.

[14] P. Dutta, Y.-S. Kuo, A. Ledeczi, T. Schmid, and P. Volgyesi. Putting the software radio on a low-calorie diet. In *Proc. ACM HotNets*, 2010.

[15] K. Harrison, S. M. Mishra, and A. Sahai. How much white-space capacity is there? In *Proc. IEEE DySpan Symp.*, 2010.

[16] A. Iyer, K. Chintalapudi, V. Navda, R. Ramjee, V. N. Padmanabhan, and C. R. Murthy. Specnet: Spectrum sensing sans frontieres. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[17] S. Kallioinen, M. Vaarakangas, P. Hui, J. Ollikainen, I. Teikari, A. Parssinen, V. Turunen, M. Kosunen, and J. Ryynanen. Multi-mode, multi-band spectrum sensor for cognitive radios embedded to a mobile phone. In *Proc. ICST CROWNCOM Conf.*, 2011.

[18] Y.-S. Kuo, P. Pannuto, T. Schmid, and P. Dutta. Reconfiguring the software radio to improve power, price, and portability. In *Proc. ACM SenSys*, 2012.

[19] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng. Towards commoditized real-time spectrum monitoring. In *Proc. ACM HotWireless*, 2014.

[20] M. e. a. Petrova. Methods and tools for estimating spectrum availability: case of single secondary user. Technical report, Ericcson AB, UKIM, KTH, RWTH, Aalto, 2012.

[21] C. Phillips, M. Ton, D. Sicker, and D. Grunwald. Practical radio environment mapping with geostatistics. In *Proc. IEEE DySpan Symp.*, 2012.

[22] S. Pollin, L. Hollevoet, F. Naessens, P. Van Wesemael, A. Dejonghe, and L. Van der Perre. Versatile sensing for mobile devices: cost, performance and hardware prototypes. In *Proc. 3rd ACM workshop on Cognitive radio networks*, 2011.

[23] S. Sen, J. Yoon, J. Hare, J. Ormont, and S. Banerjee. Can they hear me now?: a case for a client-assisted approach to monitoring wide-area wireless networks. In *Proc. ACM IMC*, pages 99–116. ACM, 2011.

[24] S. Sen, T. Zhang, M. M. Buddhikot, S. Banerjee, D. Samardzija, and S. Walker. A dual technology femto cell architecture for robust communication using whitespaces. In *Proc. IEEE DySPAN*, pages 242–253. IEEE, 2012.

[25] M. Wellens, J. Riihijarvi, M. Gordziel, and P. Mahonen. Spatial statistics of spectrum usage: From measurements to spectrum models. In *Proc. IEEE ICC*, 2009.

[26] X. Ying, J. Zhang, L. Yan, G. Zhang, M. Chenanant, and R. Chandra. Exploring indoor white spaces in metropolises. In *Proc. ACM MobiCom*, 2013.

[27] T. Zhang and S. Banerjee. A Vehicle-based Measurement Framework for Enhancing Whitespace Spectrum Databases. In *Proc. ACM MobiCom*, 2014.

[28] T. Zhang, A. Patro, N. Leng, and S. Banerjee. A wireless spectrum analyzer in your pocket. In *Proc. ACM HotMobile*, 2015.